

A7 6 compiler statement, the compiler statement being written to the temporary file in place of the second  
7 instruction;  
8 a compiler that receives the temporary file to produce an object file containing, for  
9 each of the first instructions a machine-readable object code equivalent, and for the compiler  
10 statement, the object code equivalent.

---

REMARKS

Applicant has amended the specification herein to correct obvious typographical errors and/or to improve readability. No new matter is believed to be added.

In addition, originally filed claims 1 and 2 are amended. Amendments to the claims, as well as to the specification, are delineated in the attached "Version With Markings To Show Changes Made" in which added material is shown by underlining, and deleted material is shown by inclusion in brackets [ ].

If, in the opinion of the Examiner, a telephone call would in any way expedite prosecution of this application, and invitation to call the undersigned at (415) 576-0200 is hereby extended.

Respectfully submitted,



Robert J. Bennett  
Reg. No. 27,533

TOWNSEND and TOWNSEND and CREW LLP  
Two Embarcadero Center, 8th Floor  
San Francisco, California 94111-3834  
(415) 576-0200 -- Fax (415) 576-0300  
SF 1277883 v1



VERSION WITH MARKINGS TO SHOW CHANGES MADE

Changes to the Specification:

The paragraph at page 1, lines 6-8 has been changed as follows:

The present invention relates generally to compilation of assembly language source code to **produce machine-readable** object code, and more particularly to compilation of source code having new assembly-language instructions, using an old assembler.

The paragraph at page 2, lines 12-27 has been changed as follows:

Unfortunately, this development effort usually has different groups of designers working on different aspects of the design. That is, development of the new simulator tool (ISS 14') for the new ISA is often the responsibility of the team that is also responsible for developing the new ISA. But, development of the new assembler may be **the** responsibility of a different team – often in a different geographic location, or worse, a different (third party) organization. This means that testing and debugging of the new ISA, or even the new ISS, must await completion of the new assembler. This makes it difficult for the assembler program to change quickly, much less allow it to change over a period of time as the extended ISA evolves. The developers of the extended ISA and new ISS must wait until the design and development of the new assembler is finished before using it to debug the extended ISA by compiling test programs, which may, in turn, necessitate changes in the assembler, and so on. This is a reiterative procedure that makes the overall task of changing processor/ISA designs a lengthy process. The current or old assembler is of no use in this development effort because it is incapable of properly interpreting and converting the new instructions.

The paragraphs beginning on page 4, line 13 to and including page 5, line 6 are changes as follows:

The present invention, as noted above, is a technique for employing an older assembler to produce executable object code from a source code containing old assembly language instructions, compatible with the older assembler, and added, new assembly language instructions not capable of being interpreted by the older assembler. The invention uses the data

directive feature usually found in presently available [in] assembler applications. Such data directive features are capable of taking an argument, usually in hexadecimal format, and inserting that argument in the object code unchanged.

Turning now to the figures, and for the moment Fig. 3, there is illustrated a diagrammatic representation of the method of the present invention as implemented on a processing system (not shown). As Fig. 3 shows, an original source file (File.asm) 40 contains old assembly instructions 40a (Old\_inst\_1 and Old\_instr\_2) and new assembly instructions 40b (movx.l @r1+r8, y1) that form a part of a new ISA. The original source file 40 is applied to a preprocessor (pp) software application 42. The preprocessor 42 operates to scan the source file 40, create a temporary source file 46, and write to the temporary source file 46, unchanged, the old assembly instructions 40a, 40a [40b].

Each new instruction 40b encountered by the preprocessor 42 is checked for validity and, if found to be a valid instruction, converted to its object code equivalent. That object code equivalent is then also written to the temporary source code file 46 as the argument of a data directive 41, which usually takes the form of ".DATA [data]". Thus, as Fig. 3 illustrates, the new instruction 40b, "movx.l @r1+r8, y1", is converted to its object code equivalent, "12AB" (Hex), and inserted in the temporary source code file 46 as the argument of the data directive statement 41. Each data directive statement 41 will be placed in the instruction sequence of the temporary source code file 46 at the same location (relative to the other instructions 40) corresponding to where the new instruction 40b appeared in the original source code file 40.

The paragraph at page 5, liens 12-21 has been changed as follows:

Although the old assembler is capable of converting the old instructions 40a directly to their object code equivalents, it would have been incapable of handling the new instructions 40b. However, when the old assembler 48 encounters a data directive in the source file, such as the data directive 41, it will use the argument of the data directive, **[as indicated,]** the object code equivalent of the new instruction 40b, and insert that object code equivalent in the object file 50. What appears now in the object file 50 are the machine readable object code equivalents of both the old instructions 40a of the original source code 40 and, added as data by

the data directives they were converted to, the object code equivalents of the new instructions 40b [40a].

The paragraphs at page 5, lines 25-33 have been changed as follows:

Turning now to Figs. 4A and 4B, the steps taken to assemble an assembly language program containing new and old instructions according to the invention is illustrated. Fig. 4A broadly shows the steps taken by a [the] control script (NEWASM) 44, while Fig. 4B shows the principal steps taken by the preprocessor 42.

Turning first to Fig. 4A, when the control script 44 is invoked, at step 70, it will first call the preprocessor 42 [in step 70], passing to it two arguments: the identification of the source code file 40[,], and the name of the temporary output file (File.tmp) to be created. Control is then passed to the preprocessor 42, the main operative steps of which are outlined in Fig. 4B.

Claim Amendments:

- 1                    1        **(Amended)** A method of converting a source code containing a plurality  
2 of instructions in a predetermined order, including new instructions, to object code for use by a  
3 processor, the method including the steps of:  
4                    copying plurality of instructions to a temporary file, the new instructions each  
5 being copied as data in the form of object code corresponding to such instruction; and  
6                    applying the plurality of instructions of the temporary file to an assembler to  
7 produce object code corresponding to the old instructions and the data forming object code for  
8 the new instructions.
- 1                    2.        **(Amended)** A method of assembling source code containing existing  
2 machine language instructions and new machine language instructions with an existing  
3 assembler to produce object code having machine language instructions corresponding to each  
4 the instructions of the existing instruction set and the new instructions, the method including the  
5 steps of:  
6                    copying each of the existing machine language instructions to a temporary file;

7                    copying each new machine language instruction to the temporary file as a data  
8    directive having a form corresponding to object code corresponding to such new machine  
9    language instruction; and  
10                  assembling the machine language instructions and the data directives to produce  
11    the object code.